- EXCEPTION –

Exception:

Anticipating and handling a problem that might occur due to external environmental factors. It is needed to separate small mistakes from systematic errors so that when a program is halted, it can lead the programmer to the source of the error.

1. **Grammatical errors:** Errors that occur due to a typo or a grammatically wrong command
2. **Execution error:**
   - **Logical error:** When a programmer writes an incorrect command that give bizarre results
   - **System Errors:** Errors that appear regardless of the will of the programmer
   - **Exceptions:** Errors and unexpected events that occur during program execution
     - ✓ If you divide an integer by 0
     - ✓ If something is outside the array index
     - ✓ If an improper casting occurs.
     - ✓ If you don't have a file for I/O

1. TRY/CATCH

Java treats exceptions as objects. When an exception occurs, an exception object is automatically generated.

```
try {
        trybloc: statements that are likely to cause an extension
} catch (Expection variable of type Exeption)r
        A command that will handle the extension (description of what to do when a problem occurs)
} finally {
        What to execute whether or not the extension occurred.
    }
```

2. THROWS

- If a method does not handle a checked exception, the method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature.

Throw	Throws
Used to throw the exception explicitly.	Does not throw an exception but is used to declare exceptions. This keyword is used to indicate that an exception might occur in the program or method.

```
public class Main{
   static void validate_Age(int age){
     //if specified age is < 18, throw ArithmeticException
     if(age<18)
      throw new ArithmeticException("Not eligible to vote and drive!!");
     else   //print the message
      System.out.println("Eligible to vote and drive!!");
   }
   public static void main(String args[]){
      //call validate_Age method
      validate_Age(10);
      System.out.println("rest of the code...");
   }
}
```

```
import java.io.*;
class Example_Throw {
    //declare exception using throws in the method signature
  void testMethod(int num) throws IOException, ArithmeticException{
     if(num==1)
        throw new IOException("IOException Occurred");
     else
        throw new ArithmeticException("ArithmeticException");
  }
}class Main{
  public static void main(String args[]){
   try{
        //this try block calls the above method so handle the exception
     Example_Throw obj=new Example_Throw();
     obj.testMethod(1);
    }catch(Exception ex){
      System.out.println(ex);
     }
  }
}
```

3. FREQUENTLY USED EXCEPTIONS:

- **InputMismatchException:** when input type is not correct (when an integer is inputted when the program expects a String)
- **ArithmeticException:** Thrown when an exceptional arithmetic condition has occurred (For example, an integer "divide by zero")
- **ArrayIndexOutOfBoundsException:** When a non-existing index is called whilst using an array
- **NullPointerException:** When program calls a non-existing object
- **NumberFormatException:** When program tries to non-convertible a String to a number.
- **ClassNotFoundException :** When a drive name cannot be found
- **SQLException :** when database, url, id or pw is incorrect